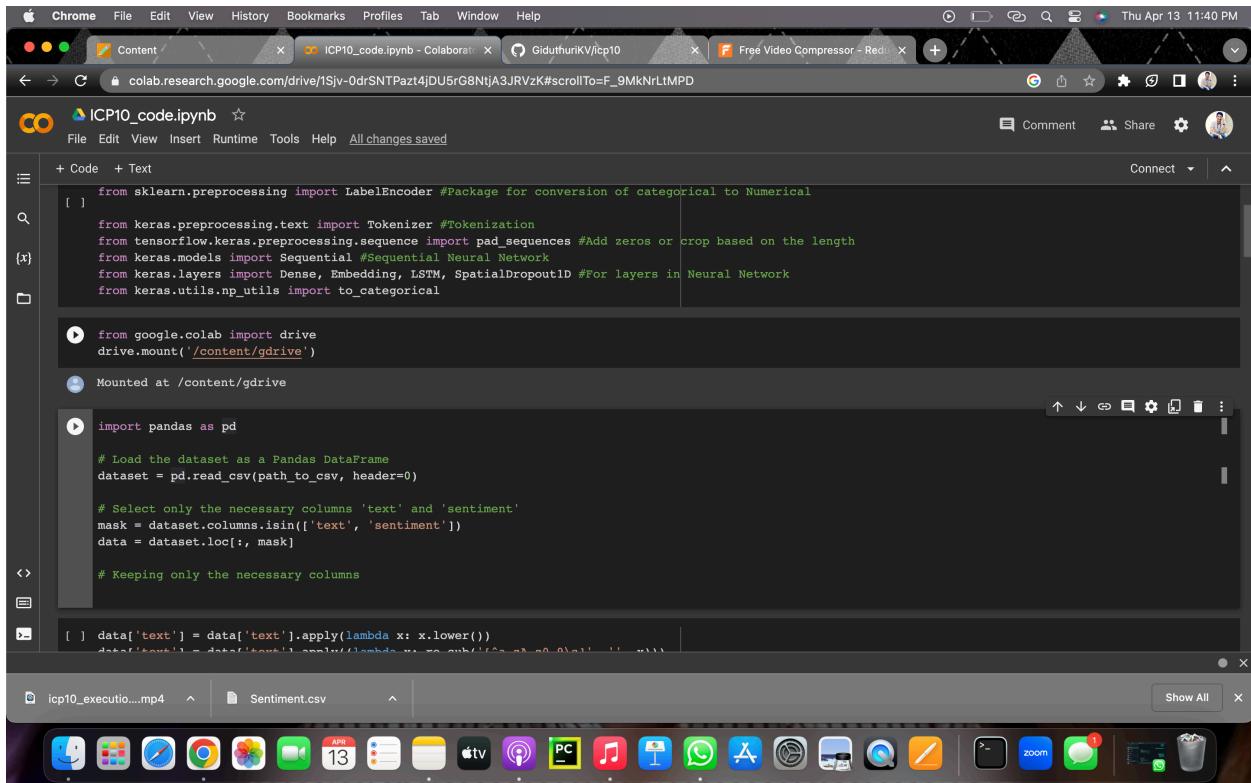


ICP 10 Report
Krishna Vamsi G
700743211

Mounted G drive



```
from sklearn.preprocessing import LabelEncoder #Package for conversion of categorical to Numerical
from keras.preprocessing.text import Tokenizer #Tokenization
from tensorflow.keras.preprocessing.sequence import pad_sequences #Add zeros or crop based on the length
from keras.models import Sequential #Sequential Neural Network
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D #For layers in Neural Network
from keras.utils.np_utils import to_categorical

from google.colab import drive
drive.mount('/content/gdrive')

import pandas as pd

# Load the dataset as a Pandas DataFrame
dataset = pd.read_csv(path_to_csv, header=0)

# Select only the necessary columns 'text' and 'sentiment'
mask = dataset.columns.isin(['text', 'sentiment'])
data = dataset.loc[:, mask]

# Keeping only the necessary columns

data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-Z0-9\s]', ' ', x))

for idx, row in data.iterrows():

    # Preprocess the text
    text = row['text'].lower()
    text = re.sub('[^a-zA-Z0-9\s]', ' ', text)
    text = text.strip()

    # Create a list of words
    words = text.split()

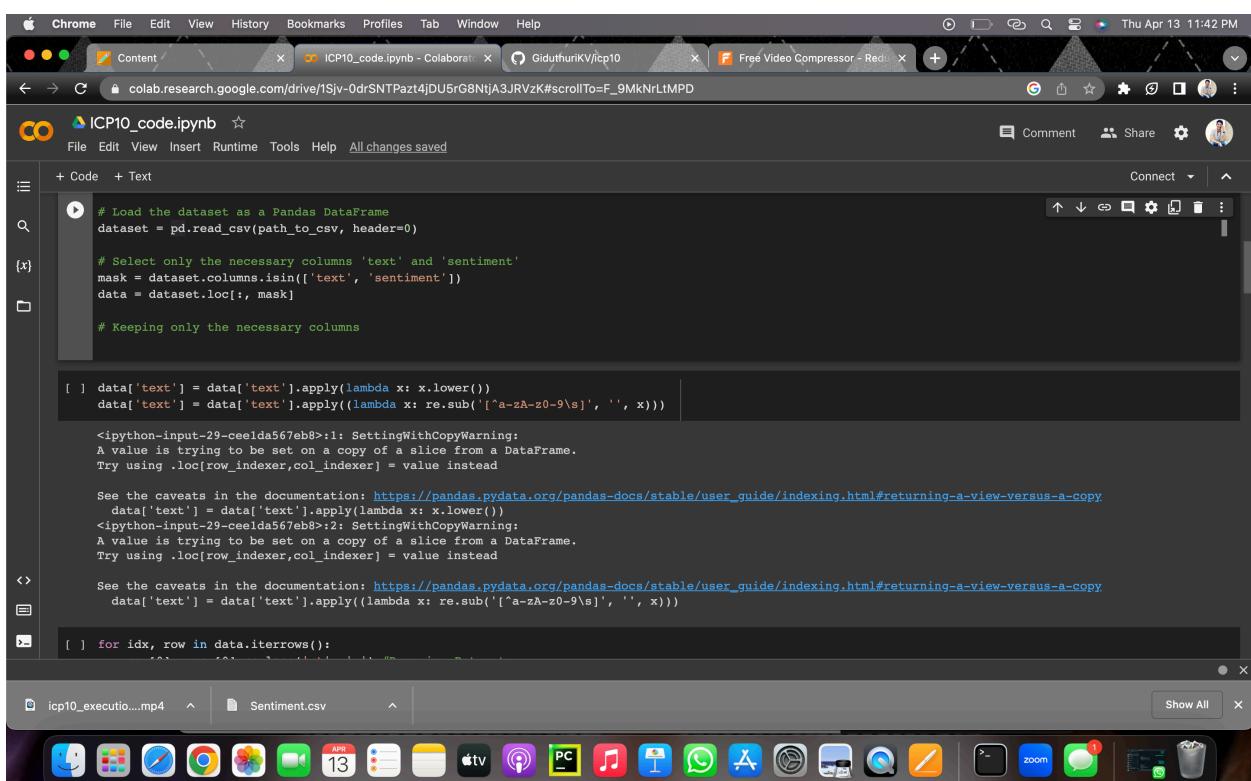
    # Create a list of word embeddings
    embeddings = [word2vec[word] if word in word2vec else np.zeros(100) for word in words]

    # Pad the embeddings
    embeddings = pad_sequences([embeddings], maxlen=maxlen, padding='post')

    # Create a list of labels
    labels = [1 if row['sentiment'] == 'positive' else 0]

    # Add the embeddings and labels to the list
    X.append(embeddings)
    y.append(labels)
```

Loaded data set as Pandas



```
# Load the dataset as a Pandas DataFrame
dataset = pd.read_csv(path_to_csv, header=0)

# Select only the necessary columns 'text' and 'sentiment'
mask = dataset.columns.isin(['text', 'sentiment'])
data = dataset.loc[:, mask]

# Keeping only the necessary columns

data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-Z0-9\s]', ' ', x))

<ipython-input-29-ceelida567eb8>:: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
data['text'] = data['text'].apply(lambda x: x.lower())
<ipython-input-29-ceelida567eb8>:: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-Z0-9\s]', ' ', x))

for idx, row in data.iterrows():

    # Preprocess the text
    text = row['text'].lower()
    text = re.sub('[^a-zA-Z0-9\s]', ' ', text)
    text = text.strip()

    # Create a list of words
    words = text.split()

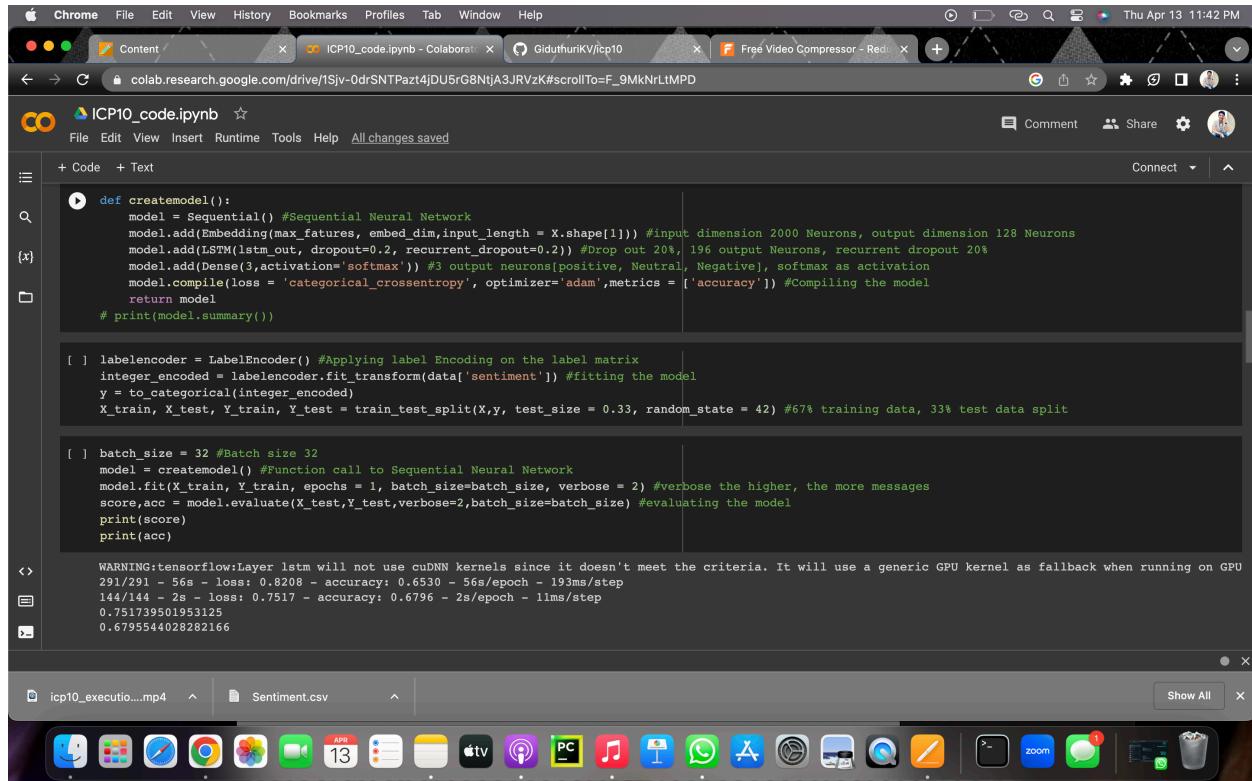
    # Create a list of word embeddings
    embeddings = [word2vec[word] if word in word2vec else np.zeros(100) for word in words]

    # Pad the embeddings
    embeddings = pad_sequences([embeddings], maxlen=maxlen, padding='post')

    # Create a list of labels
    labels = [1 if row['sentiment'] == 'positive' else 0]

    # Add the embeddings and labels to the list
    X.append(embeddings)
    y.append(labels)
```

Metrics of the model

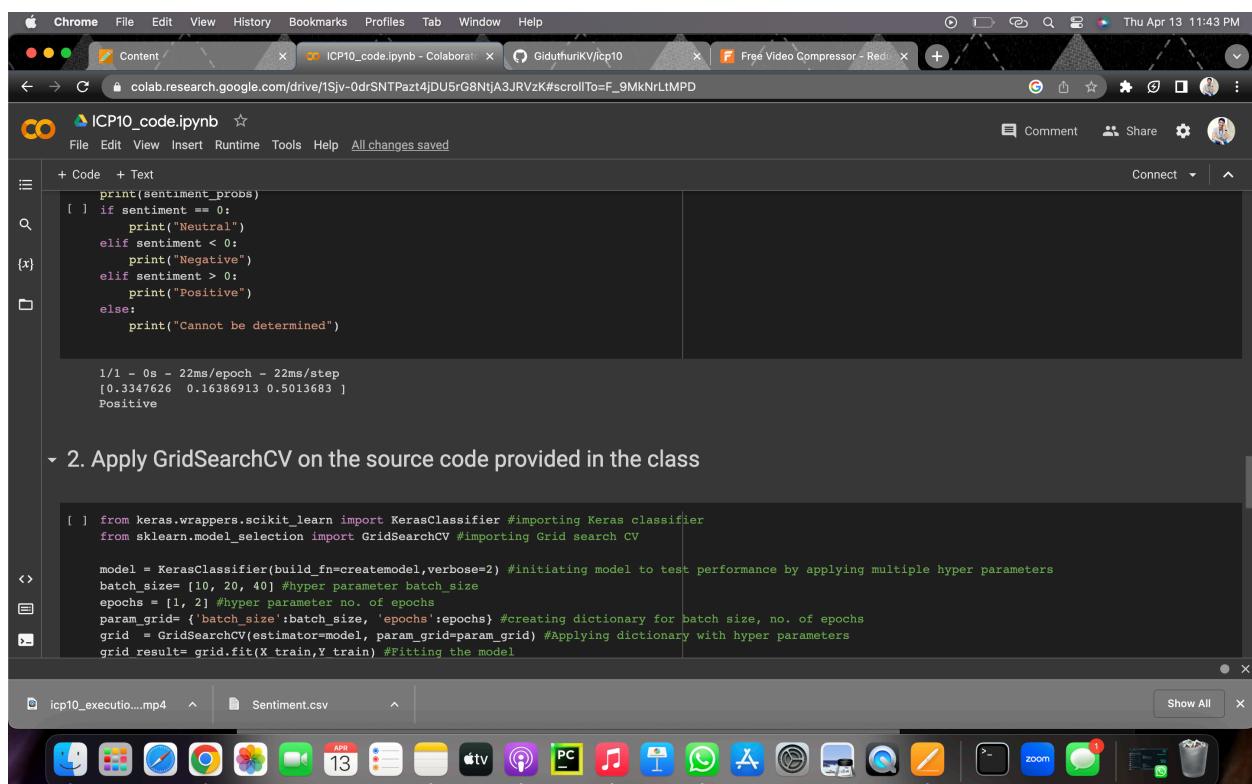


```
def createmodel():
    model = Sequential() #Sequential Neural Network
    model.add(Embedding(max_features, embed_dim, input_length = X.shape[1])) #input dimension 2000 Neurons, output dimension 128 Neurons
    model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2)) #Drop out 20%, 196 output Neurons, recurrent dropout 20%
    model.add(Dense(3,activation='softmax')) #3 output neurons[positive, Neutral, Negative], softmax as activation
    model.compile(loss = 'categorical_crossentropy', optimizer= 'adam',metrics = ['accuracy']) #Compiling the model
    return model
# print(model.summary())

labelencoder = LabelEncoder() #Applying label Encoding on the label matrix
integer_encoded = labelencoder.fit_transform(data['sentiment']) #fitting the model
y = to_categorical(integer_encoded)
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42) #67% training data, 33% test data split

batch_size = 32 #Batch size 32
model = createmodel() #Function call to Sequential Neural Network
model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2) #verbose the higher, the more messages
score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size) #evaluating the model
print(score)
print(acc)

WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU
291/291 - 56s - loss: 0.8208 - accuracy: 0.6530 - 56s/epoch - 193ms/step
144/144 - 2s - loss: 0.7517 - accuracy: 0.6796 - 2s/epoch - 11ms/step
0.751739501953125
0.6795544028282166
```



```
print(sentiment_probs)
if sentiment == 0:
    print("Neutral")
elif sentiment < 0:
    print("Negative")
elif sentiment > 0:
    print("Positive")
else:
    print("Cannot be determined")

1/1 - 0s - 22ms/epoch - 22ms/step
[0.3347626 0.16386913 0.5013683]
Positive

2. Apply GridSearchCV on the source code provided in the class

from keras.wrappers.scikit_learn import KerasClassifier #importing Keras classifier
from sklearn.model_selection import GridSearchCV #importing Grid search CV

model = KerasClassifier(build_fn=createmodel,verbose=2) #initiating model to test performance by applying multiple hyper parameters
batch_size= [10, 20, 40] #hyper parameter batch_size
epochs = [1, 2] #hyper parameter no. of epochs
param_grid= {'batch_size':batch_size, 'epochs':epochs} #creating dictionary for batch size, no. of epochs
grid = GridSearchCV(estimator=model, param_grid=param_grid) #Applying dictionary with hyper parameters
grid result= grid.fit(X_train,Y_train) #Fitting the model
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Content ICP10_code.ipynb - Colaboratory GiduthuriKV/icp10 Free Video Compressor - Redu Thu Apr 13 11:44 PM

colab.research.google.com/drive/1Sjv-0drSNTPazt4jDU5rG8NtjA3JRVzK#scrollTo=F_9MkNrLMPD

ICP10_code.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings

+ Code + Text

```
744/744 - 105s - loss: 0.8208 - accuracy: 0.6488 - 105s/epoch - 141ms/step
Epoch 2/2
744/744 - 95s - loss: 0.6808 - accuracy: 0.7127 - 95s/epoch - 127ms/step
186/186 - 3s - loss: 0.7464 - accuracy: 0.6778 - 3s/epoch - 16ms/step
WARNING:tensorflow:Layer lstm_8 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on G
Epoch 1/2
744/744 - 108s - loss: 0.8200 - accuracy: 0.6455 - 108s/epoch - 145ms/step
Epoch 2/2
744/744 - 96s - loss: 0.6682 - accuracy: 0.7186 - 96s/epoch - 130ms/step
186/186 - 2s - loss: 0.7458 - accuracy: 0.6864 - 2s/epoch - 11ms/step
WARNING:tensorflow:Layer lstm_9 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on G
Epoch 1/2
744/744 - 107s - loss: 0.8252 - accuracy: 0.6452 - 107s/epoch - 144ms/step
Epoch 2/2
744/744 - 95s - loss: 0.6764 - accuracy: 0.7123 - 95s/epoch - 128ms/step
186/186 - 2s - loss: 0.7443 - accuracy: 0.6712 - 2s/epoch - 11ms/step
WARNING:tensorflow:Layer lstm_10 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on
Epoch 1/2
744/744 - 105s - loss: 0.8182 - accuracy: 0.6490 - 105s/epoch - 141ms/step
Epoch 2/2
744/744 - 94s - loss: 0.6692 - accuracy: 0.7143 - 94s/epoch - 127ms/step
186/186 - 2s - loss: 0.7689 - accuracy: 0.6749 - 2s/epoch - 11ms/step
WARNING:tensorflow:Layer lstm_11 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on
372/372 - 61s - loss: 0.8300 - accuracy: 0.6429 - 61s/epoch - 165ms/step
93/93 - 1s - loss: 0.7640 - accuracy: 0.6606 - 1s/epoch - 12ms/step
WARNING:tensorflow:Layer lstm_12 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on
372/372 - 59s - loss: 0.8303 - accuracy: 0.6438 - 59s/epoch - 160ms/step
93/93 - 1s - loss: 0.7571 - accuracy: 0.6794 - 1s/epoch - 14ms/step
WARNING:tensorflow:Layer lstm_13 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on
372/372 - 59s - loss: 0.8337 - accuracy: 0.6450 - 59s/epoch - 158ms/step
93/93 - 1s - loss: 0.7684 - accuracy: 0.6735 - 1s/epoch - 12ms/step
```

icp10_executio....mp4 Sentiment.csv

Show All

